Cryptanalysis of a Homomorphic Encryption Scheme

Sonia Bogos*, John Gaspoz and Serge Vaudenay

EPFL
CH-1015 Lausanne, Switzerland
{soniamihaela.bogos, john.gaspoz, serge.vaudenay}@epfl.ch

Abstract. Homomorphic encryption allows to make specific operations on private data which stays encrypted. While applications such as cloud computing require to have a practical solution, the encryption scheme must be secure. In this article, we detail and analyze in-depth the homomorphic encryption scheme proposed by Zhou and Wornell in [20]. From the analysis of the encryption scheme, we are able to mount three attacks. The first attack enables to recover a secret plaintext message broadcasted to multiple users. The second attack performs a chosen ciphertext key recovery attack and it was implemented and verified. The last attack is a related chosen plaintext decryption attack.

1 Introduction

Homomorphic encryption enables operations directly in the encrypted domain. The notion of homomorphic encryption was first introduced by Rivest et al. [15] in 1978. While this was instantiated at first only for addition or multiplication, a breakthrough in this area was done by Gentry [7] that introduced the first Fully Homomorphic Encryption (FHE) scheme. An FHE scheme is one that can support computation of any arbitrary functions in the encrypted domain. While there is an extensive work done in this area [7,16,9,17,8,10,18,5,6,4,3,2,1], FHE is not yet practical and many applications do not require such a strong primitive. Instead, there are many schemes that offer a limited number of homomorphic operations. While more restrictive, these schemes may offer practical solutions depending on the application (e.g. database queries, protection of medical data).

The encryption scheme proposed in [20] is a homomorphic encryption scheme that supports three operations: addition, linear transformation and weighted inner products. Moreover, based on the three fundamental operations, the scheme enables to compute arbitrary polynomials. As

* Supported by a grant of the Swiss National Science Foundation, 200021_143899/1.

stated in [20], these operations can be useful in applications where tasks as feature extraction, recognition, classification, and data aggregation are needed. This scheme is a natural generalization of the PVW scheme [13] from binary vectors to integer vectors where techniques as ciphertext packing [1] and key switching [3] are employed to optimize the scheme. Working with plaintexts that are integer vectors (and not bits) is efficient and attractive from a practical point of view.

Our Contribution. It is desired by the cryptographic community and by many research communities (e.g. big data, medical research, cloud computing) to have a practical, secure homomorphic encryption scheme. While we believe that a lot of research should be invested in the area of homomorphic encryption, it is also essential to analyse and filter those schemes whose security is questionable.

In this article, *we first formalize the homomorphic encryption scheme from [20].* The presentation in the original paper is not very intuitive nor easy to follow. As the devil is in the details, we believe it is important to be rigorous and careful with every step of the scheme. A clear description allows us to further discover weaknesses. Thus, we are able to *mount three attacks: a broadcast encryption, a chosen ciphertext attack and a related chosen plaintext attack against this scheme.* These attacks, together with an incomplete security analysis in the original work, suggest that another approach and work is required in order to be able to construct a homomorphic scheme that works with vectors of integers.

Structure of the paper. In Section 2, we formalize the encryption scheme from [20] and describe the operations which are possible in the encrypted domain. In Section 3, we present three attacks against this encryption scheme. Finally we conclude with Section 4.

Notations. We will use small bold letters for vectors and capital bold letters for matrices (i.e $\mathbf{a}$ is a column vector). We define $|\mathbf{a}|$ to be the L_∞-norm and $\lceil a \rfloor$ the round up of a to the nearest integer. The notation $vec(\mathbf{A})$ denotes the vector that consists of all the entries in a matrix $\mathbf{A}$, where the values are taken column by column. Given a domain $\mathcal{D}$, we denote by $x \xleftarrow{U} \mathcal{D}$ the fact that x is drawn uniformly at random from $\mathcal{D}$.

2 Encryption Scheme on Integer Vectors

In this section we introduce the encryption scheme from [20] and the homomorphic operations that it supports. The presentation differs from

the original description. Our goal is to properly formalize the scheme and eliminate any ambiguity.

The scheme from [20] encrypts vectors of integers and supports three homomorphic operations that are going to be presented later in this Section.

As noted in the original work [20], the scheme relies on the idea of key switching [3]. Given a pair of public key/secret key, (pk, sk), and a different secret key sk' one is able to generate the new public key pk' that corresponds to sk'. In terms of encryption, a ciphertext c, encryption of plaintext m under the public key pk, can be switched to a new ciphertext c' that will decrypt correctly to m under the key sk'. We will formalize the scheme following the line of this concept.

We first present some methods that will ease the explanation of the encryption scheme. The first method, that we denote Bin, is taking a vector $\mathbf{c}$ where all the components are smaller than 2^ℓ. The method is constructing $\mathbf{c}^*$, the binary representation of $\mathbf{c}$, where each component of $\mathbf{c}$ is represented by ℓ bits.

Algorithm 1 $Bin(\mathbf{c}, \ell)$

Input: $n, \ell, \mathbf{c} \in \mathbb{Z}^n$ s.t. $|\mathbf{c}| < 2^\ell$
Output: $\mathbf{c}^* \in \{0,1\}^{n\ell}$

for all i with $0 \leq i \leq n-1$ **do** ▷ (i.e. each component of $\mathbf{c} = [c_0, \ldots, c_{n-1}]^T$)
    write the binary decomposition $c_i = c_{i0} + c_{i1}2 + c_{i2}2^2 + \cdots + c_{i(\ell-1)}2^{\ell-1}$
Construct a column vector $\mathbf{c}^*$ as
$\mathbf{c}^* = [c_{00}, \ldots, c_{0(\ell-1)}, c_{10}, \ldots, c_{(n-1)0}, \ldots, c_{(n-1)(\ell-1)}]^T$
return $\mathbf{c}^*$

The Dev procedure takes a matrix $\mathbf{S} \in \mathbb{Z}^{m \times n}$ and an integer ℓ, and it constructs the matrix $\mathbf{S}^* \in \mathbb{Z}^{m \times n\ell}$ such that each value S_{ij} is transformed into a block $S^*_{ij} = \left[S_{ij}, S_{ij}2, \ldots, S_{ij}2^{\ell-1}\right]$.

Given the description of these two procedures, we can state the following Lemma.

Lemma 1. *Given a matrix* $\mathbf{S} \in \mathbb{Z}^{m \times n}$*, an integer* ℓ *and an integer vector* $\mathbf{c} \in \mathbb{Z}^n$ *such that* $0 \leq |\mathbf{c}| < 2^\ell$*, we have that* $\mathbf{Sc} = Dev(\mathbf{S}, \ell) \cdot Bin(\mathbf{c}, \ell)$.

Algorithm 2 $Dev(\mathbf{S}, \ell)$

Input: $m, n, \ell, \mathbf{S} \in \mathbb{Z}^{m \times n}$
Output: $\mathbf{S}^* \in \mathbb{Z}^{m \times n\ell}$

for all i with $0 \leq i \leq m-1$ **do**
    for all j with $0 \leq j \leq n-1$ **do**
        Construct $S^*_{ij} = \left[S_{ij}, S_{ij}2, \ldots, S_{ij}2^{\ell-1}\right]$
return $\mathbf{S}^*$

Proof. We write $\mathbf{S}^* = Dev(\mathbf{S}, \ell)$ and $\mathbf{c}^* = Bin(\mathbf{c}, \ell)$. We observe that for each block of values S^*_{ij} of $\mathbf{S}^*$ and c^*_j of $\mathbf{c}^*$ we have

$$S^*_{ij}c^*_j = \left[S_{ij}\ 2S_{ij}\ \cdots\ 2^{\ell-1}S_{ij}\right] \begin{bmatrix} c_{j0} \\ c_{j1} \\ \vdots \\ c_{j(\ell-1)} \end{bmatrix}$$
$$= S_{ij}c_{j0} + 2S_{ij}c_{j1} + \ldots + 2^{\ell-1}S_{ij}c_{j(\ell-1)}$$
$$= S_{ij}(c_{j0} + 2c_{j1} + \ldots + 2^{\ell-1}c_{j(\ell-1)})$$
$$= S_{ij}c_j$$

Hence we have that $\mathbf{Sc} = \mathbf{S}^*\mathbf{c}^* = Dev(\mathbf{S}, \ell)Bin(\mathbf{c}, \ell)$. □

The three generic algorithms of the encryption scheme from [20], i.e. key generation, encryption and decryption, are described below.

2.1 Key Generation

As a first step, the user generates the public and secret keys that will be used for encryption and decryption. The key generation algorithm is described in Algorithm 3. Once the random matrices $\mathbf{T}$ and $\mathbf{A}$ and the noise matrix $\mathbf{E}$, drawn from a noise distribution χ on $\mathbb{Z}_q$, are sampled, the secret key $\mathbf{S}$ and the public key $\mathbf{M}$ are computed. The public key $\mathbf{M}$ is defined such that $\mathbf{SM} = Dev(\mathbf{I}, \ell) + \mathbf{E} \pmod q$ is verified.

Conditions imposed on the parameters ℓ, m, n, p, q, w are the ones described in Algorithm 3. In the original work [20], the authors propose a set of parameters: $\ell = 28, m = 2^7, n = 2^8, p = 2^8, q \approx 2^{50}, w = 2^{20}$.

2.2 Encryption

Given the public key $\mathbf{M}$, the public parameter w, and the plaintext $\mathbf{x} \in \mathbb{Z}^m$, the encryption algorithm outputs the ciphertext $\mathbf{c}$ that is computed as in Algorithm 4.

Algorithm 3 Key generation $Gen()$

1: **Input**: $\ell, m, n, p, q, w \in \mathbb{Z}$ where $\ell = \lceil \log_2(q-1) \rceil$, $m < n$, $q \gg p$ and $w(p-1) < q$, distribution χ
2: **Output**: $(\mathbf{M}, \mathbf{S})$, where $\mathbf{M} \in \mathbb{Z}_q^{n \times m\ell}$ and $\mathbf{S} \in \mathbb{Z}_q^{m \times n}$

3: Sample $\mathbf{T} \xleftarrow{U} \mathbb{Z}_q^{m \times (n-m)}$
4: Sample $\mathbf{A} \xleftarrow{U} \mathbb{Z}_q^{(n-m) \times m\ell}$
5: Sample $\mathbf{E} \leftarrow \chi^{m \times m\ell}$
6: Construct $\mathbf{S} = [\mathbf{I}, \mathbf{T}] \in \mathbb{Z}_q^{m \times n}$, with $\mathbf{I} \in \mathbb{Z}^{m \times m}$ as the identity matrix
7: Construct $\mathbf{M} = \begin{pmatrix} -\mathbf{TA} + Dev(\mathbf{I}, \ell) + \mathbf{E} \\ \mathbf{A} \end{pmatrix} \in \mathbb{Z}_q^{n \times m\ell}$
8: Return $(\mathbf{M}, \mathbf{S})$

Algorithm 4 Encryption algorithm $Enc(\mathbf{M}, w, \mathbf{x})$

1: **Input**: public key $\mathbf{M} \in \mathbb{Z}_q^{n \times m\ell}$, public parameter w, plaintext $\mathbf{x} \in \mathbb{Z}^m$
2: **Output**: ciphertext $\mathbf{c} \in \mathbb{Z}_q^n$

3: $\mathbf{c} = \mathbf{M} \times Bin(w\mathbf{x}, \ell) \pmod q$
4: Return $\mathbf{c}$

2.3 Decryption

At decryption, the user receives $\mathbf{c}$ and knows the value of the secret key $\mathbf{S}$ and the public parameter w. In order to decrypt, one reduces modulo q the value of $\mathbf{Sc}$. Once we change the domain from $\mathbb{Z}_q$ to $\mathbb{Z}$, by a mapping that maps $i \in \mathbb{Z}_q$ to $i \in \mathbb{Z}$, we perform division by w and round up to the nearest integer to obtain the plaintext $\mathbf{x}$.

Algorithm 5 Decryption algorithm $Dec(\mathbf{S}, w, \mathbf{c})$

1: **Input**: secret key $\mathbf{S} \in \mathbb{Z}_q^{m \times n}$, public parameter w, ciphertext $\mathbf{c} \in \mathbb{Z}_q^n$
2: **Output**: plaintext $\mathbf{x} \in \mathbb{Z}^m$

3: Compute $\mathbf{r} = \mathbf{Sc} \pmod q$
4: Change domain $\mathbb{Z}_q \to \mathbb{Z}$ to compute $\dfrac{\mathbf{r}}{w}$
5: $\mathbf{x} = \lceil \dfrac{\mathbf{r}}{w} \rfloor$
6: Return $\mathbf{x}$

Correctness. The encryption scheme described above is correct, i.e., if $\mathbf{c}$ is the encryption of a plaintext $\mathbf{x}$ as above, then one decrypts $\mathbf{c}$ to $\mathbf{x}$,

provided that he knows the secret key $\mathbf{S}$. This is possible when the noise is under a given threshold.

In order to prove the correctness of the scheme we need the following result that describes the relation that a valid encryption satisfies.

Theorem 1. *We assume that $q > w|\mathbf{x}|$ and we have the plaintext $\mathbf{x} \in \mathbb{Z}^m$. We define $\ell = \lceil \log_2(q) \rceil$. We take the distribution χ such that $x \leftarrow \chi$ is such that $x < \frac{w}{2m\ell}$ with high probability. We assume that the output of the key generation is $(\mathbf{S}, \mathbf{M})$. The vector $\mathbf{c} \in \mathbb{Z}_q^n$ is the ciphertext of $\mathbf{x}$ with length $n > m$. We have*

$$\mathbf{Sc} = q\mathbf{k} + w\mathbf{x} + \mathbf{e} \tag{1}$$

for some integer vectors $\mathbf{k}$ and $\mathbf{e}$ such that $|\mathbf{e}| < \frac{w}{2}$.

Proof. We have defined the secret key $\mathbf{S} = [\mathbf{I}, \mathbf{T}] \in \mathbb{Z}_q^{m \times n}$ and the public key

$$\mathbf{M} \equiv \begin{pmatrix} -\mathbf{TA} + Dev(\mathbf{I}, \ell) + \mathbf{E} \\ \mathbf{A} \end{pmatrix} \pmod{q}$$

We have

$$\mathbf{SM} \equiv Dev(\mathbf{I}, \ell) + \mathbf{E} \pmod{q}$$

so,

$$\mathbf{SM} = q\mathbf{k}_{\mathbf{SM}} + Dev(\mathbf{I}, \ell) + \mathbf{E}$$

with $\mathbf{k}_{\mathbf{SM}}$ an integer matrix.

From Algorithm 4, the ciphertext $\mathbf{c}$ is computed as

$$\mathbf{c} \equiv \mathbf{M}Bin(w\mathbf{x}, \ell) \pmod{q}$$

$$\mathbf{c} = q\mathbf{k}^* + \mathbf{M}Bin(w\mathbf{x}, \ell)$$

where $\mathbf{k}^*$ is an integer vector and $|\mathbf{k}^*|$ is much smaller than q when $|\mathbf{T}|$ is much smaller than q.

Finally, we have

$$\begin{aligned}
\mathbf{Sc} &= q\mathbf{Sk}^* + \mathbf{SM}Bin(w\mathbf{x}, \ell) \\
&= q\mathbf{Sk}^* + (q\mathbf{k}_{\mathbf{SM}} + Dev(\mathbf{I}, \ell) + \mathbf{E})Bin(w\mathbf{x}, \ell) \\
&= q(\mathbf{Sk}^* + \mathbf{k}_{\mathbf{SM}}Bin(w\mathbf{x}, \ell)) + w\mathbf{x} + \mathbf{E}Bin(w\mathbf{x}, \ell) \\
&= q\mathbf{k} + w\mathbf{x} + \mathbf{E}Bin(w\mathbf{x}, \ell) \\
&= q\mathbf{k} + w\mathbf{x} + \mathbf{e}
\end{aligned}$$

with $Dev(I, \ell)Bin(w\mathbf{x}, \ell) = w\mathbf{x}$ (by Lemma 1), $\mathbf{k} = \mathbf{S}\mathbf{k}^* + \mathbf{k}_{\mathbf{SM}}Bin(w\mathbf{x}, \ell)$ integer vector and $\mathbf{e} = \mathbf{E}Bin(w\mathbf{x}, \ell)$ the noise vector. $\mathbf{E}$ is sampled from distribution χ and we have $|\mathbf{E}Bin(w\mathbf{x}, \ell)| < \frac{w}{2}$. This is a required condition on equation (1) as this boundary will have an important role during the decryption. □

We can now prove the correctness of the scheme.

Lemma 2 (Correctness). *We assume that $q > w|\mathbf{x}|$ and that $(\mathbf{M}, \mathbf{S})$ is a pair of public-secret key. We assume $\mathbf{c}$ is a valid encryption of the plaintext $\mathbf{x}$ under key $\mathbf{M}$, i.e. it satisfies the relation $\mathbf{S}\mathbf{c} = q\mathbf{k} + w\mathbf{x} + \mathbf{e}$ with $|\mathbf{e}| < \frac{w}{2}$. Then $\mathbf{c}$ decrypts correctly to $\mathbf{x}$ under key $\mathbf{S}$.*

Proof. If we follow the steps of the decryption algorithm we can see that by applying the modulo on $\mathbf{S}\mathbf{c}$ we remove the $q\mathbf{k}$ value. Since we assume that $w|\mathbf{x}| < q$, the value $w\mathbf{x}$ does not get modified by the modulo operation. We have

$$\mathbf{S}\mathbf{c} \equiv w\mathbf{x} + \mathbf{e} \pmod{q}$$

After the division in $\mathbb{Z}$ we obtain

$$\frac{\mathbf{S}\mathbf{c}}{w} = \mathbf{x} + \frac{\mathbf{e}}{w}$$

By performing the round up to the nearest integer we obtain

$$\lceil \frac{\mathbf{S}\mathbf{c}}{w} \rfloor = \lceil \mathbf{x} + \frac{\mathbf{e}}{w} \rfloor = \lceil \mathbf{x} \rfloor + \lceil \frac{\mathbf{e}}{w} \rfloor = \mathbf{x}$$

Since $|\mathbf{e}| < \frac{w}{2}$, we have that

$$\frac{e_j}{w} < \frac{w}{2w} \Rightarrow e_j < \frac{1}{2}, \ \forall\ 1 \leq j \leq m$$

Thus, the decryption algorithm correctly recovers the plaintext $\mathbf{x}$. □

2.4 Key-Switching Technique

As aforementioned, the encryption scheme relies on the concept of key-switching. We are given two secret keys $\mathbf{S}, \mathbf{S}' \in \mathbb{Z}_q^{m \times n}$. The ciphertext $\mathbf{c} \in \mathbb{Z}_q^n$ decrypts to the plaintext $\mathbf{x} \in \mathbb{Z}_p^m$ under the key $\mathbf{S}$. We would like to compute a new matrix $\mathbf{M}'$, that will produce a new ciphertext $\mathbf{c}'$ such that the secret key $\mathbf{S}'$ will decrypt $\mathbf{c}'$ to the same $\mathbf{x}$.

We describe below the two methods that perform this task.

The matrix $\mathbf{M}'$ is generated such that

$$\mathbf{S}'\mathbf{M}' = Dev(\mathbf{S}, \ell) + \mathbf{E}' \pmod{q}$$

Algorithm 6 $SwitchS(\mathbf{S}, \mathbf{S}')$

1: **Input**: $\mathbf{S} = [\mathbf{I}, \mathbf{T}], \mathbf{S}' = [\mathbf{I}, \mathbf{T}']$, distribution χ
2: **Output**: $\mathbf{M}'$

3: Sample $\mathbf{A}' \xleftarrow{U} \mathbb{Z}_q^{(n-m)\times m\ell}$
4: Sample $\mathbf{E}' \leftarrow \chi^{m\times m\ell}$
5: Construct $\mathbf{M}' = \begin{pmatrix} -\mathbf{T}'\mathbf{A}' + Dev(\mathbf{S}, \ell) + \mathbf{E}' \\ \mathbf{A}' \end{pmatrix} \in \mathbb{Z}_q^{n\times m\ell}$
6: Return $\mathbf{M}'$

Algorithm 7 $SwitchC(\mathbf{c}, \mathbf{M}')$

1: **Input**: ℓ, public key $\mathbf{M}'$, ciphertext $\mathbf{c}$
2: **Output**: ciphertext $\mathbf{c}'$

3: $\mathbf{c}' = \mathbf{M}' \times Bin(\mathbf{c}, \ell) \pmod q$
4: Return c'

Lemma 3. *Let* $\mathbf{c}$ *be a valid encryption of the plaintext* $\mathbf{x}$*. Let* $\mathbf{M}' \leftarrow SwitchS(\mathbf{S}, \mathbf{S}')$ *and* $\mathbf{c}' \leftarrow SwitchC(\mathbf{c}, \mathbf{M}')$*. Then we have that* $Dec(\mathbf{S}', \mathbf{c}') = Dec(\mathbf{S}, \mathbf{c}) = \mathbf{x}$ *given that* $|\mathbf{Sc} - w\mathbf{x} \bmod q| + n\ell|E'| \leq \frac{w}{2}$.

Proof. We have

$$\begin{aligned}
\mathbf{S}'\mathbf{c}' &= \mathbf{S}'\mathbf{M}' \times Bin(\mathbf{c}, \ell) \pmod q \\
&= (Dev(\mathbf{S}, \ell) + \mathbf{E}') \times Bin(\mathbf{c}, \ell) \pmod q \\
&= Dev(\mathbf{S}, \ell)Bin(\mathbf{c}, \ell) + \mathbf{E}' \times Bin(\mathbf{c}, \ell) \pmod q \\
&= \mathbf{Sc} + \mathbf{E}' \times Bin(\mathbf{c}, \ell) \pmod q \\
&= w\mathbf{x} + \mathbf{e_1} + \mathbf{E}' \times Bin(\mathbf{c}, \ell) \pmod q \\
&= w\mathbf{x} + \mathbf{e}' \pmod q
\end{aligned}$$

where $\mathbf{Sc} = w\mathbf{x} + \mathbf{e_1} \pmod q$, $\mathbf{e}' = \mathbf{e_1} + \mathbf{E}' \times Bin(\mathbf{c}, \ell)$. We see that the noise has been increased.

In order to have $Dec(\mathbf{S}', \mathbf{c}') = Dec(\mathbf{S}, \mathbf{c})$, we need $|\mathbf{e_1}| + n\ell|\mathbf{E}'| < \frac{w}{2}$. □

Using the key-switching technique we can rephrase the key generation and the encryption algorithms as:

$Gen() \rightarrow \{\mathbf{S}, \mathbf{M}\}$
  $\mathbf{S} = [\mathbf{I}, \mathbf{T}]$
  $\mathbf{M} = SwitchS(\mathbf{I}, \mathbf{S})$
$Enc(\mathbf{M}, w, \mathbf{x}) \rightarrow \mathbf{c}$
  $\mathbf{c} = SwitchC(w\mathbf{x}, \mathbf{M})$

2.5 Operations on Encrypted Data

Three types of fundamental operations on integer vectors can be performed based on the encryption scheme from [20]: addition, linear transformation and weighted inner products. We assume that all the plaintext values are between zero and $\lfloor \frac{q}{w} \rfloor$ in order to avoid integer overflows.

Addition Let $\mathbf{c_1}$ and $\mathbf{c_2}$ be the two ciphertexts of the integer vectors $\mathbf{x_1}$ and $\mathbf{x_2}$, respectively. The addition operation $\mathbf{x_1} + \mathbf{x_2}$ is straightforward if $\mathbf{c_1}$ and $\mathbf{c_2}$ have the same secret key $\mathbf{S}$. In this situation we have that $\mathbf{c'} = \mathbf{c_1} + \mathbf{c_2} \pmod q$ is the encryption of $\mathbf{x_1} + \mathbf{x_2}$.

In the case $\mathbf{c_1}$ and $\mathbf{c_2}$ do not have the same key, we need to switch one secret key to the other.

To guarantee a valid decryption after an addition in the encrypted domain, we need to have $|\mathbf{e_1} + \mathbf{e_2}| < \frac{w}{2}$.

Linear Transformation The linear transformation $\mathbf{Gx_1}$ follows the observation that

$$\mathbf{GSc_1} = q\mathbf{Gk_1} + w\mathbf{Gx_1} + \mathbf{Ge_1}$$

So if $|\mathbf{G}|$ is much smaller than q, we can treat $\mathbf{c'} = \mathbf{c_1}$ as the ciphertext of $\mathbf{Gx_1}$ with secret key $\mathbf{GS}$ and error $\mathbf{Ge_1}$. Hence to perform the linear transformation we have to compute $Dec(\mathbf{GS}, \mathbf{c'}) = \mathbf{Gx}$.

The resulting noise after the transformation and the key-switching will be $\mathbf{Ge_1} = \mathbf{GE} \times Bin(w\mathbf{x}, \ell)$. Hence, to ensure a valid operation we must have $|\mathbf{GE} \times Bin(w\mathbf{x}, \ell)| < \frac{w}{2}$.

Weighted Inner Products Given two plaintexts, $\mathbf{x_1}$ and $\mathbf{x_2}$, encrypted as $\mathbf{c_1}$ and $\mathbf{c_2}$ with the keys $\mathbf{S_1}$ and $\mathbf{S_2}$ and a matrix $\mathbf{H}$, we can compute the weighted inner products $\mathbf{x_1}^T\mathbf{Hx_2}$.

Let $\mathbf{S'} = vec(\mathbf{S_1^T HS_2})^T$ be the new secret key, and let $\mathbf{c'} = \lceil \frac{vec(\mathbf{c_1 c_2^T})}{w} \pmod q \rfloor$ be the new ciphertext. By decrypting $\mathbf{c'}$ with the secret key $\mathbf{S'}$, we compute the weighted inner products $\mathbf{x}_1^T\mathbf{H}x_2$. The proof of this result is presented in [19,20].

Again, in order for the decryption to work, we need to ensure that the noise level after this operation in under the $\frac{w}{2}$ threshold.

As presented in [20] these three operation combined allow to compute arbitrary polynomial on integers. Note that every operation will increase the resulting noise. Hence, only a limited number of operations can be chained.

3 Attacking the Scheme

We present in this section three attacks on the encryption scheme from [20]. These attacks were implemented and tested to certify their validity.

3.1 Attack on Broadcast Encryption

We can notice that the encryption algorithm performs the multiplication between the public key $\mathbf{M}$ and the binary representation of the vector $w\mathbf{x}$. We have that $\mathbf{M} \in \mathbb{Z}_q^{n \times m\ell}$ and $Bin(w\mathbf{x}, \ell) \in \{0,1\}^{m\ell}$. This means we have n equations in $\mathbb{Z}_q$ and $m\ell$ unknowns, where $n < m\ell$. If we have access to $\lceil \frac{m\ell}{n} \rceil$ equations $\mathbf{c_i'} = \mathbf{M_i} \times Bin(w\mathbf{x}, \ell)$, where $\mathbf{M_i}$ and $\mathbf{c_i'}$ are different at each equation but the same $Bin(w\mathbf{x}, \ell)$ is used, then we could solve the system of equations by Gaussian elimination and recover the value of $\mathbf{x}$.

Let us assume we have a network of more than $\lceil \frac{m\ell}{n} \rceil$ users where every user i has its own public key $\mathbf{M_i}$. Now, let us assume that a user, e.g. Bob, wants to broadcast a secret information $\mathbf{x}$ to the users using the fixed parameters p, q, w. Hence, Bob will compute $\mathbf{c_i'} = \mathbf{M_i} \times Bin(w\mathbf{x}, \ell)$ for all the users. By listening to the traffic, an adversary $\mathcal{A}$ could obtain all the values $\mathbf{c_i'}$ broadcasted by Bob and he would have enough information to solve the system.

Indeed, the attacker $\mathcal{A}$ could use all the gathered information to produce

$$\widetilde{\mathbf{M}} = \begin{pmatrix} \mathbf{M_1} \\ \mathbf{M_2} \\ \vdots \\ \mathbf{M_z} \end{pmatrix} \in \mathbb{Z}_q^{zn \times m\ell}$$

and

$$\widetilde{\mathbf{c}} = \begin{pmatrix} \mathbf{c_1} \\ \mathbf{c_2} \\ \vdots \\ \mathbf{c_z} \end{pmatrix} \in \mathbb{Z}_q^{m\ell}$$

where $z > \lceil \frac{m\ell}{n} \rceil$.

Hence, the attacker could solve the system $\widetilde{\mathbf{M}} \times Bin(w\mathbf{x}, \ell) = \widetilde{\mathbf{c}}$ using a Gaussian elimination algorithm in $\mathcal{O}((m\ell)^3)$ [1].

[1] This attack has even a lower complexity if we use optimized matrix inversion algorithms

This attack can be seen as similar to the broadcast encryption in RSA with small keys [11]. The difference is that our attack should always work as $\frac{m\ell}{n}$ should not be too large.

A valid scenario for this attack would be one where a service provider has to send an activation key to its customers. The activation key is the same for all the customers. In such case, when the service provider has to send the encrypted activation key to enough customers, an unauthorized user could recover the activation key.

3.2 Chosen Ciphertext Attack

In this attack we assume that the adversary has access to an oracle that decrypts a given ciphertext. His goal is to retrieve the secret key $\mathbf{S}$.

We recall that the secret key is of the form $\mathbf{S} = [\mathbf{I}, \mathbf{T}] \in \mathbb{Z}_q^{m \times n}$, with $\mathbf{I} \in \mathbb{Z}^{m \times m}$ the identity matrix and $\mathbf{T} \in \mathbb{Z}_q^{m \times (n-m)}$ a random matrix. Let us define $\mathbf{T} = \begin{pmatrix} t_{a1} & t_{a2} & \dots & t_{a(n-m)} \\ t_{b1} & t_{b2} & \dots & t_{b(n-m)} \\ \vdots & \vdots & \vdots & \vdots \\ t_{m1} & t_{m2} & \dots & t_{m(n-m)} \end{pmatrix}$.

An attacker could construct $n - m$ ciphertexts $\mathbf{c_i}$, with $i \in [1, n-m]$ and where $\mathbf{c_i}$ is a vector of size n that contains zeros except a value w at the $(m+i)^{th}$ position.

With these special $\mathbf{c_i}$ we have that $\mathbf{S}\mathbf{c_i} = \begin{pmatrix} wt_{ai} \\ wt_{bi} \\ \vdots \\ wt_{mi} \end{pmatrix}$.

The attacker could ask for decryption of every $\mathbf{c_i}$ which would result in

$$Dec(\mathbf{S}, w, \mathbf{c_i}) = \mathbf{x_i} = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{im} \end{pmatrix} = \begin{pmatrix} \lceil \frac{wt_{ai}}{w} \rfloor \\ \lceil \frac{wt_{bi}}{w} \rfloor \\ \vdots \\ \lceil \frac{wt_{mi}}{w} \rfloor \end{pmatrix}$$

The second equality holds if $w|\mathbf{T}| < q$. As described in [20] the matrix $\mathbf{T}$ is chosen such that $|T| \ll w$ and that $w < q$. Thus, with a high probability we have that $w|\mathbf{T}| < q$. The decryption algorithm will output the vector

$$\mathbf{x_i} = \begin{pmatrix} \lceil \frac{wt_{ai}}{w} \rfloor \\ \lceil \frac{wt_{bi}}{w} \rfloor \\ \vdots \\ \lceil \frac{wt_{mi}}{w} \rfloor \end{pmatrix} = \begin{pmatrix} t_{ai} \\ t_{bi} \\ \vdots \\ t_{mi} \end{pmatrix}$$

Hence, by asking for the decryption of the $n - m$ $\mathbf{c_i}$ ciphertexts, we can recover the $\mathbf{T}$ matrix, hence the entire secret key $\mathbf{S}$. The attack has a complexity of $(m - n)$ requests to the decryption oracle.

3.3 Chosen Related Plaintext Attack

In order to describe the following attack, we first introduce the notions we need.

Definition 1. *Given two integers a and b, we define*

$$\mathsf{carry}(a, b) = (a + b) \oplus a \oplus b$$

where $\oplus$ denotes the bitwise exclusive OR.

We similarly define $\mathsf{carry}(a, b)$ for integer vectors component-wise.

Lemma 4. *Given two integers a and b, where their binary decomposition is $a = a_0 + a_1 * 2 + \ldots + a_n 2^n$ and $b = b_0 + b_1 * 2 + \ldots + b_n 2^n$, we have*

$$(a + b)_i = a_i + b_i + \mathsf{carry}(a, b)_i - 2 \cdot \mathsf{carry}(a, b)_{i+1}.$$

Proof. The binary addition is defined as follows:

Algorithm 8 Binary addition

1: $c_0 = 0$
2: **for all** $i = 0$ to n **do**
3: $\quad x = a_i + b_i + c_i$
4: $\quad (a + b)_i = x \bmod 2$
5: $\quad c_{i+1} = \lfloor \frac{x}{2} \rfloor$

So, by induction we have $c_i = \mathsf{carry}(a, b)_i$ at each iteration. □

Lemma 5. *Given two integer vectors* $\mathbf{x}$ *and* $\mathbf{y}$*, we have*

$$Bin(\mathbf{x}+\mathbf{y},\ell) = Bin(\mathbf{x},\ell) + Bin(\mathbf{y},\ell) + (I-2J)\cdot Bin(\mathsf{carry}(\mathbf{x},\mathbf{y}),\ell),$$

where $|\mathbf{x}|, |\mathbf{y}| < 2^\ell$*,* I *is the identity matrix and* J *is the block diagonal matrix having blocks with a diagonal of* 1 *just over the main diagonal, i.e.*

$$\begin{pmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 \\ 0 & \cdots & 0 & 0 & 0 \end{pmatrix}$$

Theorem 2. *Given two integer vectors* $\mathbf{x}$ *and* $\mathbf{y}$*, we have*

$$Enc(\mathbf{x}+\mathbf{y}) = Enc(\mathbf{x}) + Enc(\mathbf{y}) + M(I-2J)Bin(\mathsf{carry}(w\mathbf{x}, w\mathbf{y}),\ell),$$

where M *is the public key and* I *and* J *are defined in Lemma 5.*

We recall that the encryption scheme from [20] is homomorphic for the addition. This means that we expect to have $Enc(\mathbf{x}+\mathbf{y}) = Enc(\mathbf{x}) + Enc(\mathbf{y})$. But, since the ciphertext space is larger than the plaintext one, several ciphertexts map to the same plaintext. We have that $M(I-2J)Bin(\mathsf{carry}(w\mathbf{x}, w\mathbf{y}),\ell)$ is an encryption of the zero vector. Thus the relation from above is valid and it can be translated to $Enc(\mathbf{x}+\mathbf{y}) = Enc(\mathbf{x}) + Enc(\mathbf{y}) + Enc(\mathbf{0})$.

In the attack we propose, we assume there is a secret $\mathbf{x}$ and that the adversary can obtain $Enc(\mathbf{x}+\mathbf{y})$ for many chosen $\mathbf{y}$ values. The purpose is to recover $\mathbf{x}$. Interestingly, the adversary will take advantage in getting the encryption of $\mathbf{x}+\mathbf{y}$ which is not $Enc(\mathbf{x}) + Enc(\mathbf{y})$. In clear, we will see that the $y \to M(I-2J)Bin(\mathsf{carry}(w\mathbf{x}, w\mathbf{y}),\ell)$ function leaks.

In the scheme we analyse here, the parameter w is not so large. So, its Hamming weight is small. Let us denote it by $\nu = HW(w)$. For example, in [20] we have $w = 2^{20}$, so $\nu = 1$. We assume that w is an odd multiple of 2^λ of weight ν.

First, we precompute the list $\mathcal{L}$ of all possible $c = \mathsf{carry}(u, w2^i)$, for every possible i. We note that the size of $\mathcal{L}$ is bounded by ℓ^ν.

For each i and j, we set $\mathbf{t_{i,j}} = [0,\ldots,0,2^i,0,\ldots,0]$ (where 2^i is at position j). Then, we insert c in the list $T_{i,j}[M(I-2J)Bin(\mathbf{ct_{0,j}},\ell)] = c$ for all $c \in \mathcal{L}$. We obtain many tables $T_{i,j}$.

Having $Enc(\mathbf{x})$, $Enc(\mathbf{t_{i,j}})$ and $Enc(\mathbf{x}+\mathbf{t_{i,j}})$ and the hash tables $T_{i,j}$, we can obtain the value

$$\gamma_{i,j} = M(I-2J)Bin(\mathsf{carry}(w\mathbf{x}, w\mathbf{t_{i,j}}),\ell).$$

If all the elements of $T_{i,j}[\gamma_{i,j}]$ end with $\lambda + i + 2$ zero bits, we deduce that the $(\lambda + i + 1)$th least significant bit of wx_j is zero. If all elements of $T_{i,j}[\gamma_{i,j}]$ end with a bit 1 followed by $\lambda + i + 1$ zero bits, we deduce that the $(\lambda + i + 1)$th least significant bit of wx_j is one. In other cases, we cannot conclude, but these cases are unlikely to occur.

Note that the λ first least significant bits of wx_j are all zero.

By repeating this for all i and j, we recover the bits of $w\mathbf{x}$ and thus recover the secret $\mathbf{x}$.

This attack can easily be improved to reduce the number of chosen related plaintexts and the complexity.

Implementation. To experiment with the encryption scheme and to test these three attacks, we developed an implementation of the scheme in Matlab. The implementation offers all the high level functions described in Section 2 such as $Gen()$, $Dev(\mathbf{S}, \ell)$, $Enc(\mathbf{M}, w, \mathbf{x})$, etc., which enables an easy use of the encryption scheme. We implemented the three attacks. For the chosen ciphertext attack we tested the attack with several parameters, including those provided in [20]. Our tests certify that we can recover the secret key. We ran tests for the third attack and we were able to recover the unknown plaintext.

4 Conclusion

In this article we have formalized and analysed the scheme from [20]. The analysis of the homomorphic encryption scheme showed some weaknesses which resulted into three attacks. The attacks rely on realistic scenarios and enable an attacker to retrieve sensitive information such as the plaintext or the secret key.

References

1. Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in lwe-based homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2013.
2. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 309–325. ACM, 2012.
3. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Ostrovsky [12], pages 97–106.

4. Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335. Springer, 2013.
5. Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 487–504. Springer, 2011.
6. Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In Pointcheval and Johansson [14], pages 446–464.
7. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.
8. Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Ostrovsky [12], pages 107–109.
9. Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.
10. Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In Pointcheval and Johansson [14], pages 465–482.
11. Johan Hastad. On using rsa with low exponent in a public key network. In *Lecture Notes in Computer Sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 403–408, New York, NY, USA, 1986. Springer-Verlag New York, Inc.
12. Rafail Ostrovsky, editor. *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. IEEE Computer Society, 2011.
13. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.
14. David Pointcheval and Thomas Johansson, editors. *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*. Springer, 2012.
15. R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.
16. Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography - PKC 2010, 13th International Conference on*

Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.
17. Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, 2014.
18. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.
19. A. Yu, W. Lok Lai, and J. Payor. Efficient integer vector homomorphic encryption. https://courses.csail.mit.edu/6.857/2015/files/yu-lai-payor.pdf, 2015.
20. Hongchao Zhou and Gregory W. Wornell. Efficient homomorphic encryption on integer vectors and its applications. In *2014 Information Theory and Applications Workshop, ITA 2014, San Diego, CA, USA, February 9-14, 2014*, pages 1–9. IEEE, 2014.